Activity Report 2018

Project-Team PROSECCO

Programming securely with cryptography

RESEARCH CENTER
Paris

THEME
Security and Confidentiality

Table of contents

Project-Team PROSECCO

Creation of the Team: 2012 January 01, updated into Project-Team: 2012 July 01

Keywords:

Computer Science and Digital Science:

A1.1. - Architectures
A1.1.8. - Security of architectures
A1.2. - Networks
A1.2.8. - Network security
A1.3. - Distributed Systems
A2. - Software
A2.1. - Programming Languages
A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.7. - Distributed programming
A2.1.11. - Proof languages
A2.2. - Compilation
A2.2.1. - Static analysis
A2.2.5. - Run-time systems
A2.4. - Formal method for verification, reliability, certification
A2.4.2. - Model-checking
A2.4.3. - Proofs
A2.5. - Software engineering
A4. - Security and privacy
A4.3. - Cryptography
A4.3.3. - Cryptographic protocols
A4.5. - Formal methods for security
A4.6. - Authentication
A4.8. - Privacy-enhancing technologies

Other Research Topics and Application Domains:

B6. - IT and telecom
B6.1. - Software industry
B6.1.1. - Software engineering
B6.3. - Network functions
B6.3.1. - Web
B6.3.2. - Network protocols
B6.4. - Internet of things
B9. - Society and Knowledge
B9.10. - Privacy

1. Team, Visitors, External Collaborators

Research Scientists

Karthikeyan Bhargavan [Team leader, Inria, Senior Researcher, HDR]
Amal Ahmed [Inria, Advanced Research Position, until Jul 2018]
Bruno Blanchet [Inria, Senior Researcher, HDR]
Harry Halpin [Inria, Starting Research Position]
Catalin Hritcu [Inria, Researcher]
Prasad Naldurg [Inria, Advanced Research Position, from May 2018]
Éric Tanter [Inria, Advanced Research Position, from Jul 2018]

Post-Doctoral Fellows
Danel Ahman [Inria, until Sep 2018]
Roberto Blanco Martinez [Inria]

PhD Students
Benjamin Beurdouche [Inria]
Nadim Kobeissi [Inria, until Jul 2018]
Natalia Kulatova [Inria]
Kenji Maillard [Ecole Normale Supérieure Paris]
Denis Merigoux [Inria, from Nov 2018]
Marina Polubelova [Inria]
Jean-Karim Zinzindohoué [Ministère de l'Intérieur, until Jun 2018]
Carmine Abate [Inria, from Jun 2018]
Benjamin Lipp [Inria, from Jun 2018]
Jérémy Thibault [Inria, from Aug 2018]

Technical staff
Iness Ben Guirat [Inria, from Aug 2018]
Victor Dumitrescu [Inria, until Nov 2018]
Florian Groult [Inria, from Nov 2018]
Théo Laurent [Inria, from Jul 2018]
Denis Merigoux [Inria, from Feb 2018 until Jun 2018]
Marc Sylvestre [Inria, until Sep 2018]

Interns
Florian Groult [Inria, from Apr 2018 until Oct 2018]
Elizabeth Labrada Deniz [Inria, from Oct 2018]
Guido Martinez [Inria, from Sep 2018]
Carmine Abate [Inria, until May 2018]
Benjamin Lipp [Inria, until May 2018]
Jérémy Thibault [Inria, from Feb 2018 until Jul 2018]

Administrative Assistants
Anna Bednarik [Inria, until Oct 2018]
Mathieu Mourey [Inria]

Visiting Scientist
Aaron Weiss [Northeastern University, until Jul 2018]

External Collaborators
David Baelde [Ecole Normale Supérieure Cachan, until Aug 2018]
Théo Laurent [Ecole Normale Supérieure Paris, until Jun 2018]
Jean-Karim Zinzindohoué [Ministère de l'Intérieur, from Jul 2018]
Victor Dumitrescu [Nomadic Labs, from Dec 2018]

2. Overall Objectives

2.1. Programming securely with cryptography

In recent years, an increasing amount of sensitive data is being generated, manipulated, and accessed online, from bank accounts to health records. Both national security and individual privacy have come to rely on the security of web-based software applications. But even a single design flaw or implementation bug in an application may be exploited by a malicious criminal to steal, modify, or forge the private records of innocent users. Such *attacks* are becoming increasingly common and now affect millions of users every year.

The risks of deploying insecure software are too great to tolerate anything less than mathematical proof, but applications have become too large for security experts to examine by hand, and automated verification tools do not scale. Today, there is not a single widely-used web application for which we can give a proof of security, even against a small class of attacks. In fact, design and implementation flaws are still found in widely-distributed and thoroughly-vetted security libraries designed and implemented by experts.

Software security is in crisis. A focused research effort is needed if security programming and analysis techniques are to keep up with the rapid development and deployment of security-critical distributed applications based on new cryptographic protocols and secure hardware devices. The goal of our team PROSECCO is to draw upon our expertise in cryptographic protocols and program verification to make decisive contributions in this direction.

Our vision is that, over its lifetime, PROSECCO will contribute to making the use of formal techniques when programming with cryptography as natural as the use of a software debugger. To this end, our long-term goals are to design and implement programming language abstractions, cryptographic models, verification tools, and verified security libraries that developers can use to deploy provably secure distributed applications. Our target applications include cryptographic protocol implementations, hardware-based security APIs, smartphone- and browser-based web applications, and cloud-based web services. In particular, we aim to verify the full application: both the cryptographic core and the high-level application code. We aim to verify implementations, not just models. We aim to account for computational cryptography, not just its symbolic abstraction.

We identify five key focus areas for our research in the short- to medium term.

2.1.1. New programming languages for verified software

Building realistic verified applications requires new programming languages that enable the systematic development of efficient software hand-in-hand with their proofs of correctness. Our current focus is on designing and implementing the programming language F*, in collaboration with Microsoft Research. F* (pronounced F star) is an ML-like functional programming language aimed at program verification. Its type system includes polymorphism, dependent types, monadic effects, refinement types, and a weakest precondition calculus. Together, these features allow expressing precise and compact specifications for programs, including functional correctness and security properties. The F* type-checker aims to prove that programs meet their specifications using a combination of SMT solving and manual proofs. Programs written in F* can be translated to efficient OCaml, F#, or C for execution. The main ongoing use case of F* is building a verified, drop-in replacement for the whole HTTPS stack in Project Everest (a larger collaboration with Microsoft Research). This includes verified implementations of TLS 1.2 and 1.3 and of the underlying cryptographic primitives.

2.1.2. Symbolic verification of cryptographic applications

We aim to develop our own security verification tools for models and implementations of cryptographic protocols and security APIs using symbolic cryptography. Our starting point is the tools we have previously developed: the specialized cryptographic prover ProVerif, the reverse engineering and formal test tool Tookan, and the F* verification system. These tools are already used to verify industrial-strength cryptographic protocol implementations and commercial cryptographic hardware. We plan to extend and combine these approaches to capture more sophisticated attacks on applications consisiting of protocols, software, and hardware, as well as to prove symbolic security properties for such composite systems.

2.1.3. Computational verification of cryptographic applications

We aim to develop our own cryptographic application verification tools that use the computational model of cryptography. The tools include the computational prover CryptoVerif, and the F* verification system. Working together, we plan to extend these tools to analyze, for the first time, cryptographic protocols, security APIs, and their implementations under fully precise cryptographic assumptions. We also plan to pursue links between symbolic and computational verification, such as computational soundness results that enable computational proofs by symbolic techniques.

2.1.4. Efficient formally secure compilers for tagged architectures

We aim to leverage emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilation chains for realistic low-level programming languages (the C language, and Low* a safe subset of C embedded in F* for verification). These compilation chains will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilation chains target a tagged architecture, which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules.

2.1.5. Building provably secure web applications

We aim to develop analysis tools and verified libraries to help programmers build provably secure web applications. The tools will include static and dynamic verification tools for client- and server-side JavaScript web applications, their verified deployment within HTML5 websites and browser extensions, as well as type-preserving compilers from high-level applications written in F* to JavaScript. In addition, we plan to model new security APIs in browsers and smartphones and develop the first formal semantics for various HTML5 web standards. We plan to combine these tools and models to analyze the security of multi-party web applications, consisting of clients on browsers and smartphones, and servers in the cloud.

3. Research Program

3.1. Symbolic verification of cryptographic applications

Despite decades of experience, designing and implementing cryptographic applications remains dangerously error-prone, even for experts. This is partly because cryptographic security is an inherently hard problem, and partly because automated verification tools require carefully-crafted inputs and are not widely applicable. To take just the example of TLS, a widely-deployed and well-studied cryptographic protocol designed, implemented, and verified by security experts, the lack of a formal proof about all its details has regularly led to the discovery of major attacks (including several in PROSECCO) on both the protocol and its implementations, after many years of unsuspecting use.

As a result, the automated verification for cryptographic applications is an active area of research, with a wide variety of tools being employed for verifying different kinds of applications.

In previous work, we have developed the following three approaches:

- ProVerif: a symbolic prover for cryptographic protocol models
- Tookan: an attack-finder for PKCS#11 hardware security devices
- F*: a new language that enables the verification of cryptographic applications

3.1.1. *Verifying cryptographic protocols with ProVerif*

Given a model of a cryptographic protocol, the problem is to verify that an active attacker, possibly with access to some cryptographic keys but unable to guess other secrets, cannot thwart security goals such as authentication and secrecy [70]; it has motivated a serious research effort on the formal analysis of cryptographic protocols, starting with [65] and eventually leading to effective verification tools, such as our tool ProVerif.

To use ProVerif, one encodes a protocol model in a formal language, called the applied pi-calculus, and ProVerif abstracts it to a set of generalized Horn clauses. This abstraction is a small approximation: it just ignores the number of repetitions of each action, so ProVerif is still very precise, more precise than, say, tree automata-based techniques. The price to pay for this precision is that ProVerif does not always terminate; however, it terminates in most cases in practice, and it always terminates on the interesting class of *tagged protocols* [60]. ProVerif can handle a wide variety of cryptographic primitives, defined by rewrite rules or by some equations, and prove a wide variety of security properties: secrecy [58], [44], correspondences (including authentication) [59], and observational equivalences [57]. Observational equivalence means that an adversary cannot distinguish two processes (protocols); equivalences can be used to formalize a wide range of properties, but they are particularly difficult to prove. Even if the class of equivalences that ProVerif can prove is limited to equivalences between processes that differ only by the terms they contain, these equivalences are useful in practice and ProVerif has long been the only tool that proves equivalences for an unbounded number of sessions. (Maude-NPA in 2014 and Tamarin in 2015 adopted ProVerif's approach to proving equivalences.)

Using ProVerif, it is now possible to verify large parts of industrial-strength protocols, such as TLS [52], Signal [68], JFK [45], and Web Services Security [56], against powerful adversaries that can run an unlimited number of protocol sessions, for strong security properties expressed as correspondence queries or equivalence assertions. ProVerif is used by many teams at the international level, and has been used in more than 120 research papers (references available at http://proverif.inria.fr/proverif-users.html).

3.1.2. *Verifying security APIs using Tookan*

Security application programming interfaces (APIs) are interfaces that provide access to functionality while also enforcing a security policy, so that even if a malicious program makes calls to the interface, certain security properties will continue to hold. They are used, for example, by cryptographic devices such as smartcards and Hardware Security Modules (HSMs) to manage keys and provide access to cryptographic functions whilst keeping the keys secure. Like security protocols, their design is security critical and very difficult to get right. Hence formal techniques have been adapted from security protocols to security APIs.

The most widely used standard for cryptographic APIs is RSA PKCS#11, ubiquitous in devices from smartcards to HSMs. A 2003 paper highlighted possible flaws in PKCS#11 [62], results which were extended by formal analysis work using a Dolev-Yao style model of the standard [63]. However at this point it was not clear to what extent these flaws affected real commercial devices, since the standard is underspecified and can be implemented in many different ways. The Tookan tool, developed by Steel in collaboration with Bortolozzo, Centenaro and Focardi, was designed to address this problem. Tookan can reverse engineer the particular configuration of PKCS#11 used by a device under test by sending a carefully designed series of PKCS#11 commands and observing the return codes. These codes are used to instantiate a Dolev-Yao model of the device's API. This model can then be searched using a security protocol model checking tool to find attacks. If an attack is found, Tookan converts the trace from the model checker into the sequence of PKCS#11 queries needed to make the attack and executes the commands directly on the device. Results obtained by Tookan are remarkable: of 18 commercially available PKCS#11 devices tested, 10 were found to be susceptible to at least one attack.

3.1.3. *Verifying cryptographic applications using F**

Verifying the implementation of a protocol has traditionally been considered much harder than verifying its model. This is mainly because implementations have to consider real-world details of the protocol, such as message formats, that models typically ignore. This leads to a situation that a protocol may have been proved secure in theory, but its implementation may be buggy and insecure. However, with recent advances in both program verification and symbolic protocol verification tools, it has become possible to verify fully functional protocol implementations in the symbolic model. One approach is to extract a symbolic protocol model from an implementation and then verify the model, say, using ProVerif. This approach has been quite successful, yielding a verified implementation of TLS in F# [55]. However, the generated models are typically quite large and whole-program symbolic verification does not scale very well.

An alternate approach is to develop a verification method directly for implementation code, using well-known program verification techniques. Our current focus is on designing and implementing the programming language F* [73], [49], in collaboration with Microsoft Research. F* (pronounced F star) is an ML-like functional programming language aimed at program verification. Its type system includes polymorphism, dependent types, monadic effects, refinement types, and a weakest precondition calculus. Together, these features allow expressing precise and compact specifications for programs, including functional correctness and security properties. The F* type-checker aims to prove that programs meet their specifications using a combination of SMT solving and manual proofs. Programs written in F* can be translated to efficient OCaml, F#, or C for execution [71]. The main ongoing use case of F* is building a verified, drop-in replacement for the whole HTTPS stack in Project Everest [53] (a larger collaboration with Microsoft Research). This includes a verified implementation of TLS 1.2 and 1.3 [54].

3.2. Computational verification of cryptographic applications

Proofs done by cryptographers in the computational model are mostly manual. Our goal is to provide computer support to build or verify these proofs. In order to reach this goal, we have designed the automatic tool CryptoVerif, which generates proofs by sequences of games. We already applied it to important protocols such as TLS [52] and Signal [68] but more work is still needed in order to develop this approach, so that it is easier to apply to more protocols. We also design and implement techniques for proving implementations of protocols secure in the computational model. In particular, CryptoVerif can generate implementations from CryptoVerif specifications that have been proved secure [61]. We plan to continue working on this approach.

A different approach is to directly verify cryptographic applications in the computational model by typing. A recent work [66] shows how to use refinement typechecking in F7 to prove computational security for protocol implementations. In this method, henceforth referred to as computational F7, typechecking is used as the main step to justify a classic game-hopping proof of computational security. The correctness of this method is based on a probabilistic semantics of F# programs and crucially relies on uses of type abstraction and parametricity to establish strong security properties, such as indistinguishability.

In principle, the two approaches, typechecking and game-based proofs, are complementary. Understanding how to combine these approaches remains an open and active topic of research.

An alternative to direct computation proofs is to identify the cryptographic assumptions under which symbolic proofs, which are typically easier to derive automatically, can be mapped to computational proofs. This line of research is sometimes called computational soundness and the extent of its applicability to real-world cryptographic protocols is an active area of investigation.

3.3. F*: A Higher-Order Effectful Language for Program Verification

F* [73], [49] is a verification system for effectful programs developed collaboratively by Inria and Microsoft Research. It puts together the automation of an SMT-backed deductive verification tool with the expressive power of a proof assistant based on dependent types. After verification, F* programs can be extracted to efficient OCaml, F#, or C code [71]. This enables verifying the functional correctness and security of realistic applications. F*'s type system includes dependent types, monadic effects, refinement types, and a weakest precondition calculus. Together, these features allow expressing precise and compact specifications for programs, including functional correctness and security properties. The F* type-checker aims to prove that programs meet their specifications using a combination of SMT solving and interactive proofs. The main ongoing use case of F* is building a verified, drop-in replacement for the whole HTTPS stack in Project

Everest. This includes verified implementations of TLS 1.2 and 1.3 [54] and of the underlying cryptographic primitives [74].

3.4. Efficient Formally Secure Compilers to a Tagged Architecture

Severe low-level vulnerabilities abound in today's computer systems, allowing cyber-attackers to remotely gain full control. This happens in big part because our programming languages, compilers, and architectures were designed in an era of scarce hardware resources and too often trade off security for efficiency. The semantics of mainstream low-level languages like C is inherently insecure, and even for safer languages, establishing security with respect to a high-level semantics does not guarantee the absence of low-level attacks. Secure compilation using the coarse-grained protection mechanisms provided by mainstream hardware architectures would be too inefficient for most practical scenarios.

We aim to leverage emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilation chains for realistic low-level programming languages (the C language, and Low* a safe subset of C embedded in F* for verification [71]). These compilation chains will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilation chains target a tagged architecture [50], which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules. We hope to experimentally evaluate and carefully optimize the efficiency of our secure compilation chains on realistic workloads and standard benchmark suites. We are also using property-based testing and formal verification to provide high confidence that our compilation chains are indeed secure. Formally, we are constructing machine-checked proofs of a new security criterion we call robustly safe compilation, which is defined as the preservation of safety properties even against an adversarial context [46], [47]. This strong criterion complements compiler correctness and ensures that no machine-code attacker can do more harm to securely compiled components than a component already could with respect to a secure source-level semantics.

3.5. Provably secure web applications

Web applications are fast becoming the dominant programming platform for new software, probably because they offer a quick and easy way for developers to deploy and sell their *app*s to a large number of customers. Third-party web-based apps for Facebook, Apple, and Google, already number in the hundreds of thousands and are likely to grow in number. Many of these applications store and manage private user data, such as health information, credit card data, and GPS locations. To protect this data, applications tend to use an ad hoc combination of cryptographic primitives and protocols. Since designing cryptographic applications is easy to get wrong even for experts, we believe this is an opportune moment to develop security libraries and verification techniques to help web application programmers.

As a typical example, consider commercial password managers, such as LastPass, RoboForm, and 1Password. They are implemented as browser-based web applications that, for a monthly fee, offer to store a user's passwords securely on the web and synchronize them across all of the user's computers and smartphones. The passwords are encrypted using a master password (known only to the user) and stored in the cloud. Hence, no-one except the user should ever be able to read her passwords. When the user visits a web page that has a login form, the password manager asks the user to decrypt her password for this website and automatically fills in the login form. Hence, the user no longer has to remember passwords (except her master password) and all her passwords are available on every computer she uses.

Password managers are available as browser extensions for mainstream browsers such as Firefox, Chrome, and Internet Explorer, and as downloadable apps for Android and Apple phones. So, seen as a distributed application, each password manager application consists of a web service (written in PHP or Java), some number of browser extensions (written in JavaScript), and some smartphone apps (written in Java or Objective C). Each of these components uses a different cryptographic library to encrypt and decrypt password data. How do we verify the correctness of all these components?

We propose three approaches. For client-side web applications and browser extensions written in JavaScript, we propose to build a static and dynamic program analysis framework to verify security invariants. To this end, we have developed two security-oriented type systems for JavaScript, Defensive JavaScript [64] [64] and TS* [72], and used them to guarantee security properties for a number of JavaScript applications. For Android smartphone apps and web services written in Java, we propose to develop annotated JML cryptography libraries that can be used with static analysis tools like ESC/Java to verify the security of application code. For clients and web services written in F# for the .NET platform, we propose to use F* to verify their correctness. We also propose to translate verified F* web applications to JavaScript via a verified compiler that preserves the semantics of F* programs in JavaScript.

3.6. Design and Verification of next-generation protocols: identity, blockchains, and messaging

Building on our work on verifying and re-designing pre-existing protocols like TLS and Web Security in general, with the resources provided by the NEXTLEAP project, we are working on both designing and verifying new protocols in rapidly emerging areas like identity, blockchains, and secure messaging. These are all areas where existing protocols, such as the heavily used OAuth protocol, are in need of considerable re-design in order to maintain privacy and security properties. Other emerging areas, such as blockchains and secure messaging, can have modifications to existing pre-standard proposals or even a complete 'clean slate' design. As shown by Prosecco's work, newer standards, such as IETF OAuth, W3C Web Crypto, and W3C Web Authentication API, can have vulnerabilities fixed before standardization is complete and heavily deployed. We hope that the tools used by Prosecco can shape the design of new protocols even before they are shipped to standards bodies. We have seen considerable progress in identity with the UnlimitID design and with messaging via the IETF MLS effort, with new work on blockchain technology underway.

4. Application Domains

4.1. Cryptographic Protocol Libraries

Cryptographic protocols such as TLS, SSH, IPSec, and Kerberos are the trusted base on which the security of modern distributed systems is built. Our work enables the analysis and verification of such protocols, both in their design and implementation. Hence, for example, we build and verify models and reference implementations for well-known protocols such as TLS and SSH, as well as analyze their popular implementations such as OpenSSL.

4.2. Hardware-based security APIs

Cryptographic devices such as Hardware Security Modules (HSMs) and smartcards are used to protect long-terms secrets in tamper-proof hardware, so that even attackers who gain physical access to the device cannot obtain its secrets. These devices are used in a variety of scenarios ranging from bank servers to transportation cards (e.g. Navigo). Our work investigates the security of commercial cryptographic hardware and evaluates the APIs they seek to implement.

4.3. Web application security

Web applications use a variety of cryptographic techniques to securely store and exchange sensitive data for their users. For example, a website may serve pages over HTTPS, authenticate users with a single sign-on protocol such as OAuth, encrypt user files on the server-side using XML encryption, and deploy client-side cryptographic mechanisms using a JavaScript cryptographic library. The security of these applications depends on the public key infrastructure (X.509 certificates), web browsers' implementation of HTTPS and the same origin policy (SOP), the semantics of JavaScript, HTML5, and their various associated security standards, as well as the correctness of the specific web application code of interest. We build analysis tools to find bugs in all these artifacts and verification tools that can analyze commercial web applications and evaluate their security against sophisticated web-based attacks.

5. Highlights of the Year

5.1. Highlights of the Year

- We published 20 papers at top-tier conferences and journals such as POPL (5), ICFP (2), PLDI (1), OOPSLA (1), ACM CCS (1), IEEE S&P (1), IEEE CSF (1), TOPLAS (1), and JCS (1).
- The HACL* verified cryptographic library developed in our group was integrated by Linux (WireGuard) and Tezos, and more verified crypto primitives were integrated in Mozilla Firefox.
- We organized a Dagstuhl Seminar on Secure Compilation (18201)
- Catalin Hritcu served as Program Chair for the Workshop on Principles of Secure Compilation at POPL'18

6. New Software and Platforms

6.1. Cryptosense Analyzer

SCIENTIFIC DESCRIPTION: Cryptosense Analyzer (formerly known as Tookan) is a security analysis tool for cryptographic devices such as smartcards, security tokens and Hardware Security Modules that support the most widely-used industry standard interface, RSA PKCS#11. Each device implements PKCS#11 in a slightly different way since the standard is quite open, but finding a subset of the standard that results in a secure device, i.e. one where cryptographic keys cannot be revealed in clear, is actually rather tricky. Cryptosense Analyzer analyses a device by first reverse engineering the exact implementation of PKCS#11 in use, then building a logical model of this implementation for a model checker, calling a model checker to search for attacks, and in the case where an attack is found, executing it directly on the device. It has been used to find at least a dozen previously unknown flaws in commercially available devices.
FUNCTIONAL DESCRIPTION: Cryptosense Analyzer (formerly known as Tookan) is a security analysis tool for cryptographic devices such as smartcards,

- Participants: Graham Steel and Romain Bardou
- Contact: Graham Steel
- URL: https://cryptosense.com/

6.2. CryptoVerif

Cryptographic protocol verifier in the computational model
KEYWORDS: Security - Verification - Cryptographic protocol
FUNCTIONAL DESCRIPTION: CryptoVerif is an automatic protocol prover sound in the computational model. In this model, messages are bitstrings and the adversary is a polynomial-time probabilistic Turing machine. CryptoVerif can prove secrecy and correspondences, which include in particular authentication. It provides a generic mechanism for specifying the security assumptions on cryptographic primitives, which can handle in particular symmetric encryption, message authentication codes, public-key encryption, signatures, hash functions, and Diffie-Hellman key agreements. It also provides an explicit formula that gives the probability of breaking the protocol as a function of the probability of breaking each primitives, this is the exact security framework.
NEWS OF THE YEAR: Bruno Blanchet modified ProVerif and CryptoVerif to improve the compatibility between these two tools (see the section on ProVerif). This feature is released in CryptoVerif 2.00.

Bruno Blanchet implemented several extensions of CryptoVerif, in particular: 1) reworked the model of Diffie-Hellman key agreements, in particular to account for the absence of public key validation in popular Diffie-Hellman groups like Curve25519, which is used in many modern protocols, 2) support for the proof of indistinguishability between two games given by the user, 3) facilitate the interactive proofs. Program points, used for instance to insert case distinctions, can now be designated as the line that matches a regular expression, instead of using a number. This is much more stable in case the protocol model is slightly modified. Groups of variables can be designated as all variables that match a regular expression. These features are not released yet.

- Participants: Bruno Blanchet and David Cadé
- Contact: Bruno Blanchet
- Publications: Composition Theorems for CryptoVerif and Application to TLS 1.3 - Composition Theorems for CryptoVerif and Application to TLS 1.3 - Proved Implementations of Cryptographic Protocols in the Computational Model - Proved Generation of Implementations from Computationally Secure Protocol Specifications - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols - Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach
- URL: http://cryptoverif.inria.fr/

6.3. F*

FStar
KEYWORDS: Programming language - Software Verification
FUNCTIONAL DESCRIPTION: F* is a new higher order, effectful programming language (like ML) designed with program verification in mind. Its type system is based on a core that resembles System Fw (hence the name), but is extended with dependent types, refined monadic effects, refinement types, and higher kinds. Together, these features allow expressing precise and compact specifications for programs, including functional correctness properties. The F* type-checker aims to prove that programs meet their specifications using an automated theorem prover (usually Z3) behind the scenes to discharge proof obligations. Programs written in F* can be translated to OCaml, F#, or JavaScript for execution.

- Participants: Antoine Delignat-Lavaud, Catalin Hritcu, Cédric Fournet, Chantal Keller, Karthikeyan Bhargavan and Pierre-Yves Strub
- Contact: Catalin Hritcu
- URL: https://www.fstar-lang.org/

6.4. miTLS

KEYWORDS: Cryptographic protocol - Software Verification
FUNCTIONAL DESCRIPTION: miTLS is a verified reference implementation of the TLS protocol. Our code fully supports its wire formats, ciphersuites, sessions and connections, re-handshakes and resumptions, alerts and errors, and data fragmentation, as prescribed in the RFCs, it interoperates with mainstream web browsers and servers. At the same time, our code is carefully structured to enable its modular, automated verification, from its main API down to computational assumptions on its cryptographic algorithms.

- Participants: Alfredo Pironti, Antoine Delignat-Lavaud, Cédric Fournet, Jean-Karim Zinzindohoué, Karthikeyan Bhargavan, Pierre-Yves Strub and Santiago Zanella-Béguelin
- Contact: Karthikeyan Bhargavan
- URL: https://github.com/mitls/mitls-fstar

6.5. ProVerif

KEYWORDS: Security - Verification - Cryptographic protocol
FUNCTIONAL DESCRIPTION: ProVerif is an automatic security protocol verifier in the symbolic model (so called Dolev-Yao model). In this model, cryptographic primitives are considered as black boxes. This protocol verifier is based on an abstract representation of the protocol by Horn clauses. Its main features are:

It can verify various security properties (secrecy, authentication, process equivalences).

It can handle many different cryptographic primitives, specified as rewrite rules or as equations.

It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space.
NEWS OF THE YEAR: Marc Sylvestre extended his interactive simulator of protocols modeled in ProVerif to simulate the semantics of biprocesses, used to prove observational equivalence between two processes. He also made minor improvements to this simulator and to the graphical display of attacks.

Bruno Blanchet modified ProVerif and CryptoVerif to improve the compatibility between these two tools. It is now possible for simple examples to use the same input file with both tools, for instance to try to find attacks in the symbolic model using ProVerif, and if no attack is found, then prove the protocol in the computational model using CryptoVerif. For more complex examples, the differences between the files to provide for each tool are considerably reduced. The cryptographic primitives are specified in distinct libraries, one for ProVerif and one for CryptoVerif, because the assumptions on primitives are very different in the symbolic and computational models. These features are released in ProVerif 2.00.

Vincent Cheval and Bruno Blanchet implemented several extensions of ProVerif: 1) support for integer counters, with incrementation and inequality tests, 2) lemmas and axioms to give intermediate results to ProVerif, which it exploits to help proving subsequent queries, by deriving additional information in the Horn clauses that it uses to perform the proofs, 3) proofs by induction on the length of the trace, by giving as lemma the property to prove, but obviously for strictly shorter traces. These features are not released yet.

- Participants: Bruno Blanchet, Marc Sylvestre and Vincent Cheval
- Contact: Bruno Blanchet
- Publications: Automated reasoning for equivalences in the applied pi calculus with barriers - Automated Reasoning for Equivalences in the Applied Pi Calculus with Barriers - Automated reasoning for equivalences in the applied pi calculus with barriers - Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif - Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols
- URL: http://proverif.inria.fr/

6.6. HACL*

High Assurance Cryptography Library
KEYWORDS: Cryptography - Software Verification
FUNCTIONAL DESCRIPTION: HACL* is a formally verified cryptographic library in F*, developed by the Prosecco team at Inria Paris in collaboration with Microsoft Research, as part of Project Everest.

HACL stands for High-Assurance Cryptographic Library and its design is inspired by discussions at the HACS series of workshops. The goal of this library is to develop verified C reference implementations for popular cryptographic primitives and to verify them for memory safety, functional correctness, and secret independence.

- Contact: Karthikeyan Bhargavan
- URL: https://github.com/mitls/hacl-star

7. New Results

7.1. Composition Theorems for CryptoVerif and Application to TLS 1.3

Participant: Bruno Blanchet.

We presented composition theorems for security protocols, to compose a key exchange protocol and a symmetric-key protocol that uses the exchanged key. Our results rely on the computational model of cryptography and are stated in the framework of the tool CryptoVerif. They support key exchange protocols that guarantee injective or non-injective authentication. They also allow random oracles shared between the composed protocols. To our knowledge, they are the first composition theorems for key exchange stated for a computational protocol verification tool, and also the first to allow such flexibility.

As a case study, we applied our composition theorems to a proof of TLS 1.3 Draft-18. This work fills a gap in our previous analysis of TLS 1.3 in CryptoVerif [52]. It appears in [31], [39].

7.2. Mechanised Cryptographic Proof of the WireGuard VPN Protocol

Participants: Benjamin Lipp, Bruno Blanchet, Karthikeyan Bhargavan.

WireGuard is a free and open source Virtual Private Network (VPN) that aims to replace IPsec and OpenVPN. It is based on a new cryptographic protocol derived from the Noise Protocol Framework. We provide the first mechanised cryptographic proof of the protocol underlying WireGuard, using the CryptoVerif proof assistant.

We analyse the entire WireGuard protocol as it is, including transport data messages, in an ACCE-style model. We contribute proofs for correctness, message secrecy, forward secrecy, mutual authentication, session uniqueness, and resistance against key compromise impersonation, identity mis-binding, and replay attacks. We also discusse the strength of the identity hiding provided by WireGuard.

Our work also provides novel theoretical contributions that are reusable beyond WireGuard. First, we extend CryptoVerif to account for the absence of public key validation in popular Diffie-Hellman groups like Curve25519, which is used in many modern protocols including WireGuard. To our knowledge, this is the first mechanised cryptographic proof for any protocol employing such a precise model. Second, we prove several indifferentiability lemmas that are useful to simplify the proofs for sequences of key derivations. This work is under submission.

7.3. Meta-F*: Proof automation with SMT, Tactics, and Metaprograms

Participants: Guido Martinez, Danel Ahman, Victor Dumitrescu, Nick Giannarakis [Princeton University], Chris Hawblitzel [Microsoft Research], Catalin Hritcu, Monal Narasimhamurthy [University of Colorado Boulder], Zoe Paraskevopoulou [Princeton University], Clément Pit-Claudel [MIT], Jonathan Protzenko [Microsoft Research], Tahina Ramananandro [Microsoft Research], Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research].

We introduced Meta-F* [69], a tactics and metaprogramming framework for the F* program verifier. The main novelty of Meta-F* is allowing to use tactics and metaprogramming to discharge assertions not solvable by SMT, or to just simplify them into well-behaved SMT fragments. Plus, Meta-F* can be used to generate verified code automatically.

Meta-F* is implemented as an F* effect, which, given the powerful effect system of F*, heavily increases code reuse and even enables the lightweight verification of metaprograms. Metaprograms can be either interpreted, or compiled to efficient native code that can be dynamically loaded into the F* type-checker and can interoperate with interpreted code. Evaluation on realistic case studies shows that Meta-F* provides substantial gains in proof development, efficiency, and robustness.

7.4. When Good Components Go Bad: Formally Secure Compilation Despite Dynamic Compromise

Participants: Carmine Abate, Arthur Azevedo de Amorim [CMU], Roberto Blanco, Ana Nora Evans [University of Virginia], Guglielmo Fachini [Nozomi Networks], Catalin Hritcu, Théo Laurent, Benjamin C. Pierce [University of Pennsylvania], Marco Stronati [Nomadic Labs], Andrew Tolmach [Portland State University].

We proposed a new formal criterion [47] for evaluating secure compilation schemes for unsafe languages, expressing end-to-end security guarantees for software components that may become compromised after encountering undefined behavior—for example, by accessing an array out of bounds.

Our criterion is the first to model dynamic compromise in a system of mutually distrustful components with clearly specified privileges. It articulates how each component should be protected from all the others—in particular, from components that have encountered undefined behavior and become compromised. Each component receives secure compilation guarantees—in particular, its internal invariants are protected from compromised components—up to the point when this component itself becomes compromised, after which we assume an attacker can take complete control and use this component's privileges to attack other components. More precisely, a secure compilation chain must ensure that a dynamically compromised component cannot break the safety properties of the system at the target level any more than an arbitrary attacker-controlled component (with the same interface and privileges, but without undefined behaviors) already could at the source level.

To illustrate the model, we construct a secure compilation chain for a small unsafe language with buffers, procedures, and components, targeting a simple abstract machine with built-in compartmentalization. We give a careful proof (mostly machine-checked in Coq) that this compiler satisfies our secure compilation criterion. Finally, we show that the protection guarantees offered by the compartmentalized abstract machine can be achieved at the machine-code level using either software fault isolation or a tag-based reference monitor.

7.5. The Meaning of Memory Safety

Participants: Arthur Azevedo de Amorim [CMU], Catalin Hritcu, Benjamin C. Pierce [University of Pennsylvania].

We give a rigorous characterization of what it means for a programming language to be memory safe [51], capturing the intuition that memory safety supports local reasoning about state. We formalize this principle in two ways. First, we show how a small memory-safe language validates a noninterference property: a program can neither affect nor be affected by unreachable parts of the state. Second, we extend separation logic, a proof system for heap-manipulating programs, with a memory-safe variant of its frame rule. The new rule is stronger because it applies even when parts of the program are buggy or malicious, but also weaker because it demands a stricter form of separation between parts of the program state. We also consider a number of pragmatically motivated variations on memory safety and the reasoning principles they support. As an application of our characterization, we evaluate the security of a previously proposed dynamic monitor for memory safety of heap-allocated data.

7.6. Recalling a Witness: Foundations and Applications of Monotonic State

Participants: Danel Ahman, Cédric Fournet [Microsoft Research], Catalin Hritcu, Kenji Maillard, Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research].

We provide a way to ease the verification of programs whose state evolves monotonically [48]. The main idea is that a property witnessed in a prior state can be soundly recalled in the current state, provided (1) state evolves according to a given preorder, and (2) the property is preserved by this preorder. In many scenarios, such monotonic reasoning yields concise modular proofs, saving the need for explicit program invariants. We distill our approach into the monotonic-state monad, a general yet compact interface for Hoare-style reasoning about monotonic state in a dependently typed language. We prove the soundness of the monotonic-state monad and use it as a unified foundation for reasoning about monotonic state in the F* verification system. Based on this foundation, we build libraries for various mutable data structures like monotonic references and apply these libraries at scale to the verification of several distributed applications.

7.7. A Monadic Framework for Relational Verification: Applied to Information Security, Program Equivalence, and Optimizations

Participants: Niklas Grimm [Vienna University of Technology], Kenji Maillard, Cédric Fournet [Microsoft Research], Catalin Hritcu, Matteo Maffei [Vienna University of Technology], Jonathan Protzenko [Microsoft Research], Tahina Ramananandro [Microsoft Research], Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research], Santiago Zanella-Béguelin [Microsoft Research].

Relational properties describe multiple runs of one or more programs. They characterize many useful notions of security, program refinement, and equivalence for programs with diverse computational effects, and they have received much attention in the recent literature. Rather than developing separate tools for special classes of effects and relational properties, we advocate using a general purpose proof assistant as a unifying framework for the relational verification of effectful programs. The essence of our approach is to model effectful computations using monads and to prove relational properties on their monadic representations, making the most of existing support for reasoning about pure programs [67].

We apply this method in F* and evaluate it by encoding a variety of relational program analyses, including information flow control, program equivalence and refinement at higher order, correctness of program optimizations and game-based cryptographic security. By relying on SMT-based automation, unary weakest preconditions, user-defined effects, and monadic reification, we show that, compared to unary properties, verifying relational properties requires little additional effort from the F* programmer.

7.8. A Formal Treatment of Accountable Proxying over TLS

Participants: Karthikeyan Bhargavan, Ioana Boureanu [University of Surrey], Antoine Delignat-Lavaud [Microsoft Research], Pierre-Alain Fouque [University of Rennes], Cristina Onete [University of Limoges].

Much of Internet traffic nowadays passes through active proxies, whose role is to inspect, filter, cache, or transform data exchanged between two endpoints. To perform their tasks, such proxies modify channel-securing protocols, like TLS, resulting in serious vulnerabilities. Such problems are exacerbated by the fact that middleboxes are often invisible to one or both endpoints, leading to a lack of accountability. A recent protocol, called mcTLS, pioneered accountability for proxies, which are authorized by the endpoints and given limited read/write permissions to application traffic.

Unfortunately, we show that mcTLS is insecure: the protocol modifies the TLS protocol, exposing it to a new class of middlebox-confusion attacks. Such attacks went unnoticed mainly because mcTLS lacked a formal analysis and security proofs. Hence, our second contribution is to formalize the goal of accountable proxying over secure channels. Third, we propose a provably-secure alternative to soon-to-be-standardized mcTLS: a generic and modular protocol-design that carefully composes generic secure channel-establishment protocols, which we prove secure. Finally, we present a proof-of-concept implementation of our design, instantiated with unmodified TLS 1.3 draft 23, and evaluate its overheads [29].

7.9. hacspec: towards verifiable crypto standards

Participants: Karthikeyan Bhargavan, Franziskus Kiefer [Mozilla], Pierre-Yves Strub [Ecole Polytechnique].

We designed and published hacspec, a formal specification language for cryptographic primitives. Specifications (specs) written in hacspec are succinct, easy to read and implement, and lend themselves to formal verification using a variety of existing tools. The syntax of hacspec is similar to the pseudocode used in cryptographic standards but is equipped with a static type system and syntax checking tools that can find errors. Specs written in hacspec are executable and can hence be tested against test vectors taken from standards and specified in a common format. Finally, hacspec is designed to be compilable to other formal specification languages like F*, EasyCrypt, Coq, and cryptol, so that it can be used as the basis for formal proofs of functional correctness and cryptographic security using various verification frameworks.

We published a paper presenting the syntax, design, and tool architecture of hacspec. We demonstrated the use of the language to specify popular cryptographic algorithms, and developed preliminary compilers from hacspec to F* and to EasyCrypt. Our eventual goal is to invite authors of cryptographic standards to write their pseudocode in hacspec and to help the formal verification community develop the language and tools that are needed to promote high-assurance cryptographic sofware backed by mathematical proofs. All our code is released publicly on GitHub.

7.10. Largest-scale user study of secure messaging and API usage

Participants: Francesca Musiani [CNRS], Ksenia Ermoshina [CNRS], Harry Halpin, Iness Ben Guirat [INSAT].

As part of the NEXTLEAP EC project, we engaged in the largest ever user study of secure messaging applications, focusing on typical users as well as "high-risk" users in the Middle East and Ukraine, as well as developers.[41]. This work has been shared with standardization efforts such as the IETF Message Layer Security (MLS) effort in which Inria is participating, as well as W3C standardization of the W3C Web Authentication API. This work helped influence the formal verification of the privacy properties of hardware-based cryptographic authentication, which is a feature needed by many at risk users whose accounts are often the focus of hacks. This work has also led a fundamental inquiry into the social governance of standards and the role of formal verification in the future of standards.[42] As this work is highly interdisciplinary, it has featured collaboration with sociologists at CNRS and interns from INSAT in Tunisia, as well as a lecture series hosted at Centre Pompidou under the direction of Bernard Stiegler and Harry Halpin.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. AnaStaSec

Title: Static Analysis for Security Properties (ANR générique 2014.)

Other partners: Inria Paris/EPI Antique, Inria Rennes/EPI Celtique, Airbus Operations SAS, AMOSSYS, CEA-LIST, TrustInSoft

Duration: January 2015 - September 2019.

Coordinator: Jérôme Féret, EPI Antique, Inria Paris (France)

Participant: Bruno Blanchet

Abstract: The project aims at using automated static analysis techniques for verifying security and confidentiality properties of critical avionics software.

8.1.1.2. AJACS

Title: AJACS: Analyses of JavaScript Applications: Certification and Security

Other partners: Inria-Rennes/Celtique, Inria-Saclay/Toccata, Inria-Sophia Antipolis/INDES, Imperial College London

Duration: October 2014 - March 2019.

Coordinator: Alan Schmitt, Inria (France)

Participants: Karthikeyan Bhargavan, Bruno Blanchet, Nadim Kobeissi

Abstract: The goal of the AJACS project is to provide strong security and privacy guarantees for web application scripts. To this end, we propose to define a mechanized semantics of the full JavaScript language, the most widely used language for the Web, to develop and prove correct analyses for JavaScript programs, and to design and certify security and privacy enforcement mechanisms.

8.1.1.3. SafeTLS

Title: SafeTLS: La sécurisation de l'Internet du futur avec TLS 1.

Other partners: Université Rennes 1, IRMAR, Inria Sophia Antipolis, SGDSN/ANSSI

Duration: October 2016 - September 2020

Coordinator: Pierre-Alain Fouque, Université de Rennes 1 (France)

Participants: Karthikeyan Bhargavan

Abstract: Our project, SafeTLS, addresses the security of both TLS 1.3 and of TLS 1.2 as they are (expected to be) used, in three important ways: (1) A better understanding: We will provide a better understanding of how TLS 1.2 and 1.3 are used in real-world applications; (2) Empowering clients: By developing a tool that will show clients the quality of their TLS connection and inform them of potential security and privacy risks; (3) Analyzing implementations: We will analyze the soundness of current TLS 1.2 implementations and use automated verification to provide a backbone of a secure TLS 1.3 implementation.

8.1.1.4. TECAP

Title: TECAP: Protocol Analysis - Combining Existing Tools (ANR générique 2017.)

Other partners: Inria Nancy/EPI PESTO, Inria Sophia Antipolis/EPI MARELLE, IRISA, LIX, LSV - ENS Cachan.

Duration: January 2018 - December 20

Coordinator: Vincent Cheval, EPI PESTO, Inria Nancy (France)

Participants: Bruno Blanchet, Benjamin Lipp

Abstract: A large variety of automated verification tools have been developed to prove or find attacks on security protocols. These tools differ in their scope, degree of automation, and attacker models. The aim of this project is to get the best of all these tools, meaning, on the one hand, to improve the theory and implementations of each individual tool towards the strengths of the others and, on the other hand, build bridges that allow the cooperations of the methods/tools. We will focus in this project on the tools CryptoVerif, EasyCrypt, Scary, ProVerif, Tamarin, AKiSs and APTE.

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. ERC Consolidator Grant: CIRCUS

Title: CIRCUS: An end-to-end verification architecture for building Certified Implementations of Robust, Cryptographically Secure web applications

Duration: April 2016 - March 2021

Coordinator: Karthikeyan Bhargavan, Inria

The security of modern web applications depends on a variety of critical components including cryptographic libraries, Transport Layer Security (TLS), browser security mechanisms, and single sign-on protocols. Although these components are widely used, their security guarantees remain poorly understood, leading to subtle bugs and frequent attacks. Rather than fixing one attack at a time, we advocate the use of formal security verification to identify and eliminate entire classes of vulnerabilities in one go.

CIRCUS proposes to take on this challenge, by verifying the end-to-end security of web applications running in mainstream software. The key idea is to identify the core security components of web browsers and servers and replace them by rigorously verified components that offer the same functionality but with robust security guarantees.

8.2.1.2. ERC Starting Grant: SECOMP

Title: SECOMP: Efficient Formally Secure Compilers to a Tagged Architecture

Duration: Jan 2017 - December 2021

Coordinator: Catalin Hritcu, Inria

Abstract: The SECOMP project is aimed at leveraging emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilation chains for realistic low-level programming languages (the C language, and Low* a safe subset of C embedded in F* for verification). These compilation chains will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilation chains target a tagged architecture, which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules. We will use property-based testing and formal verification to provide high confidence that our compilers are indeed secure.

8.2.1.3. NEXTLEAP

Title: NEXTLEAP: NEXT generation Legal Encryption And Privacy

Programm: H2020

Duration: January 2016 - December 2018

Coordinator: Harry Halpin, Inria

Other partners: IMDEA, University College London, CNRS, IRI, and Merlinux

Abstract: NEXTLEAP aims to create, validate, and deploy protocols that can serve as pillars for a secure, trust-worthy, and privacy-respecting Internet. For this purpose NEXTLEAP will develop an interdisciplinary study of decentralisation that provides the basis on which these protocols cann be designed, working with sociologists to understand user needs. The modular specification of decentralized protocols, implemented as verified open-source software modules, will be done for both privacy-preserving secure federated identity as well as decentralized secure messaging services that hide metadata (e.g., who, when, how often, etc.).

8.3. International Initiatives

8.3.1. Inria International Partners

8.3.1.1. Informal International Partners

We have a range of long- and short-term collaborations with various universities and research labs. We summarize them by project:

- TLS analysis: Microsoft Research (Cambridge), Mozilla, University of Rennes
- F*: Microsoft Research (Redmond, Cambridge, Bangalore), MSR-Inria, CMU, MIT, University of Ljubljana, Nomadic Labs, Zen Protocol, Princeton University
- SECOMP: MPI-SWS, CISPA, Stanford University, CMU, University of Pennsylvania, Portland State University, University of Virginia, University of Iai
- Micro-Policies: University of Pennsylvania, Portland State University, MIT, Draper Labs, Dover Microsystems

8.3.2. Participation in Other International Programs

8.3.2.1. SSITH/HOPE

Title: Advanced New Hardware Optimized for Policy Enforcement, A New HOPE

Program: DARPA SSITH

Duration: December 2017 - February 2021

Coordinator: Charles Stark Draper Laboratory

Other Participants: Inria Paris, University of Pennsylvania, MIT, Portland State University, Dover Microsystems, DornerWorks

Participants from Inria Prosecco: Catalin Hritcu, Roberto Blanco, Jérémy Thibault

Abstract: A New HOPE builds on results from the Inherently Secure Processor (ISP) project that has been internally funded at Draper. Recent architectural improvements decouple the tagged architecture from the processor pipeline to improve performance and flexibility for new processors. HOPE securely maintains metadata for each word in application memory and checks every instruction against a set of installed security policies. The HOPE security architecture exposes tunable parameters that support Performance, Power, Area, Software compatibility and Security (PPASS) search space exploration. Flexible software-defined security policies cover all 7 SSITH CWE vulnerability classes, and policies can be tuned to meet PPASS requirements; for example, one can trade granularity of security checks against performance using different policy configurations. HOPE will design and formalize a new high-level domain-specific language (DSL) for defining security policies, based on previous research and on extensive experience with previous policy languages. HOPE will formally verify that installed security policies satisfy system-wide security requirements. A secure boot process enables policies to be securely updated on deployed HOPE systems. Security policies can adapt based on previously detected attacks. Over the multi-year, multi-million dollar Draper ISP project, the tagged security architecture approach has evolved from early prototypes based on results from the DARPA CRASH program towards easier integration with external designs, and is better able to scale from micro to server class implementations. A New HOPE team is led by Draper and includes faculty from University of Pennsylvania (Penn), Portland State University (PSU), Inria, and MIT, as well as industry collaborators from DornerWorks and Dover Microsystems. In addition to Draper's in-house expertise in hardware design, cyber-security (defensive and offensive, hardware and software) and formal methods, the HOPE team includes experts from all domains relevant to SSITH, including (a) computer architecture: DeHon (Penn), Shrobe (MIT); (b) formal methods including programming languages and security: Pierce (Penn), Tolmach (PSU), Hritcu (Inria); and (c) operating system integration (DornerWorks). Dover Microsystems is a spin-out from Draper that will commercialize concepts from the Draper ISP project.

8.3.2.2. Everest Expedition

Program: Microsoft Expedition and MSR-Inria Collaborative Research Project

Expedition Participants: Microsoft Research (Cambridge, Redmond, Bangalore), Inria, MSR-Inria, CMU, University of Edinburgh

Duration of current MSR-Inria Project: October 2017 – October 2020

Participants from Inria Prosecco: Karthikeyan Bhargavan, Catalin Hritcu, Danel Ahman, Benjamin Beurdouche, Victor Dumitrescu, Nadim Kobeissi, Théo Laurent, Guido Martínez, Denis Merigoux, Marina Polubelova, Jean-Karim Zinzindohoué

Participants from other Inria teams: David Pichardie (Celtique), Jean-Pierre Talpin (TEA)

Abstract: The HTTPS ecosystem (HTTPS and TLS protocols, X.509 public key infrastructure, crypto algorithms) is the foundation on which Internet security is built. Unfortunately, this ecosystem is brittle, with headline-grabbing attacks such as FREAK and LogJam and emergency patches many times a year.

Project Everest addresses this problem by constructing a high-performance, standards-compliant, formally verified implementation of components in HTTPS ecosystem, including TLS, the main protocol at the heart of HTTPS, as well as the main underlying cryptographic algorithms such as AES, SHA2 or X25519.

At the TLS level, for instance, we are developing new implementations of existing and forthcoming protocol standards and formally proving, by reduction to cryptographic assumptions on their core algorithms, that our implementations provide a secure-channel abstraction between the communicating endpoints. Implementations of the core algorithms themselves are also verified, producing performant portable C code or highly optimized assembly language.

We aim for our verified components to be drop-in replacements suitable for use in mainstream web browsers, servers, and other popular tools and are actively working with the community at large to improve the ecosystem.

https://project-everest.github.io

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Amal Ahmed (Northeastern University, USA) joined Inria as a Visiting Professor from September 2017 to July 2018; she gave a seminar on “Compositional Compiler Verification for a Multi-Language World”.
- Aaron Weiss (Northeastern University, USA) joined Inria as a Visiting Scientist from September 2017 to July 2018; he gave a seminar on “Rust Distilled: An Expressive Tower of Languages”
- Justin Hsu (University of Wisconsin–Madison, USA) visited Prosecco on 26 January 2018 and gave a talk entitled “From Couplings to Probabilistic Relational Program Logics”
- Deepak Garg (MPI-SWS, Germany) visited Prosecco on 21 February and 6 December 2018
- Marco Patrignani (CISPA, Germany) visited Prosecco on 21 February 2018
- Arthur Azevedo de Amorim (CMU) visited Prosecco on 10–13 April 2018 and gave a seminar on “The Meaning of Memory Safety”
- Prasad Naldurg (IBM Research, India) joined Prosecco as a Visiting Researcher from May 2018; he gave a Prosecco seminar on “Encrypted Analytics: Computing directly on encrypted databases”
- Vincent Gramoli (NICTA/Data61-CSIRO and University of Sydney, Australia) visited Prosecco on 27 June 2018 and gave a seminar on “The Red Belly Blockchain: Speed, Security, Scalability”
- Éric Tanter (University of Chile) joined Prosecco as Visiting Professor from July 2018 to February 2019; he gave a Prosecco seminar on “Gradual Parametricity, Revisited” and many other talks
- Andrew Tolmach (Portland State University, USA) visited Prosecco on 2–4 July 2018
- Ilya Sergey (University College London, UK) visited Prosecco on 5 September 2018 and gave a seminar on “Deductive Synthesis of Programs that Alter Data Structures”
- Jonathan Aldrich (CMU, USA) visited Prosecco on 22–26 November 2018 and gave a seminar on “Object Capabilities, Effects, and Abstraction”
- tefan Ciobâcă (University of Iai, Romania) visited Prosecco on 3–7 December 2018
- Amin Timany (KU Leuven, Belgium) visited Prosecco on 3–7 December 2018
- Cédric Fournet (Microsoft Research, UK) has visited Prosecco on various occasions
- Jonathan Protzenko (Microsoft Research, USA) has visited Prosecco on various occasions

8.4.1.1. Internships

- Benjamin Lipp (Karlsruhe Institute of Technology, Germany): from Dec 2017 to May 2018 – advised by Bruno Blanchet and Karthik Bhargavan

- Carmine Abate (University of Trento, Italy): from Dec 2017 to May 2018 – advised by Catalin Hritcu
- Jérémy Thibault (ENS Rennes, France): from Feb to Jul 2018 – advised by Catalin Hritcu
- Florian Groult (University of Orleans, France): from Apr to Oct 2018 – advised by Catalin Hritcu
- Guido Martinez (CIFASIS-CONICET Rosario, Argentina): from Sep to December 2018 – advised by Catalin Hritcu
- Elizabeth Labrada Deniz (University of Chile): from Oct 2018 to January 2019 – advised by Éric Tanter and Catalin Hritcu
- Iness Ben Guirat (INSAT): from August 2018 to January 2019 – advised by Harry Halpin

8.4.2. Visits to International Teams

- Catalin Hritcu, Danel Ahman, and Victor Dumitrescu visited Microsoft Research (Redmond, USA) on 5–25 March 2018
- Catalin Hritcu, Carmine Abate, and Jérémy Thibault visited the MPI-SWS (Saarbrücken, Germany) on 27–28 March 2018
- Catalin Hritcu visited Draper Labs (Cambridge, MA, USA) on 30 May 2018
- Karthikeyan Bhargavan, Catalin Hritcu, Danel Ahman, Benjamin Beurdouche, Victor Dumitrescu, Guido Martínez, Denis Merigoux, and Marina Polubelova visited Microsoft Research (Cambridge, UK) for Everest “All-Hands” meeting
- Harry Halpin visited the NEXTLEAP team meeting (Lausanne, Switzerland) on 15–17th of January.
- Harry Halpin visited the NEXTLEAP team meeting (Freibourg, Germany) on 21–22nd of November.
- Harry Halpin visited the final PANORAMIX team meeting (Athens, Greece) on 24–25th of September.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

- Catalin Hritcu and Amal Ahmed co-organized a Dagstuhl Seminar on Secure Compilation (18201)
- Harry Halpin and Bart Preneel co-organized the ECRYPT-CSA workshop on Crypto Policies (22-23 January 2018) in Brussels, Belgium.

9.1.1.2. Member of the Organizing Committees

- Catalin Hritcu and Amal Ahmed were organizers for PriSC 2018 and the upcoming PriSC 2019

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

- Catalin Hritcu was the PC chair for the 2nd Workshop on Principles of Secure Compilation (PriSC) at POPL 2018
- Harry Halpin was General Chair of the 1st Workshop on the Decentralization of Governance at INSCI 2018

9.1.2.2. Member of the Conference Program Committees

- Bruno Blanchet was PC member of RESSI 2018 (*Rendez-vous de la Recherche et de l’Enseignement de la Sécurité des Systèmes d’Information*).

- Catalin Hritcu was PC member of EuroS&P 2018, ESOP 2018, and CCS 2018
- Karthikeyan Bhargavan was PC member of IEEE S&P 2018, ACM CCS 2018, and POST 2018
- Harry Halpin was PC Member of SSR 2018, ACM WWW 2018, and ISWC 2018.

9.1.2.3. Reviewer

- Catalin Hritcu served as a reviewer for the Journal of Automated Reasoning (JAR)

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

Associate Editor

- of the *International Journal of Applied Cryptography (IJACT)* – Inderscience Publishers: Bruno Blanchet

9.1.4. Invited Talks

- Catalin Hritcu gave an Invited Keynote talk at the Working Formal Methods Symposium (FROM) in June 2018
- Catalin Hritcu gave invited talks at Nomadic Labs (Tezos), IRIF Verification Seminar (Paris 7), and SoSySec seminar (IRISA Rennes)
- Karthikeyan Bhargavan gave invited talks at Security Standardization Research (SSR 2018), Formal Methods and Tools for Security (FMATS 2018), Crypto Welcomes TLS 1.3 (CWTLS), and the annual GDR Securité meeting.
- Harry Halpin gave invited talks at the EPFL Summer Research Institute in July 2018, the Web 3.0 Summit in October 2018, and Binance Labs in December 2018.

9.1.5. Leadership within the Scientific Community

- Catalin Hritcu served as the Artifact Evaluation Co-Chair for POPL 2018 and POPL 2019

9.1.6. Scientific Expertise

- Bruno Blanchet is a member of the specialized temporary scientific committee of ANSM (*Agence nationale de sécurité du médicament et des produits de santé*), on the cybersecurity of software medical devices.
- Bruno Blanchet participated to a review of the code of the Tezos blockchain by the Inria Foundation (March–May 2018).
- Harry Halpin participated as a member of the advisory board to the PANORAMIX EC H2020 project (2018).

9.1.7. Research Administration

- Bruno Blanchet was co-president of the Inria hiring committee for PhD, post-docs, and *délégations* (*Commision des Emplois Scientifiques*, CES).
- Bruno Blanchet was representative of Inria Paris at the DIM RFSI (*Domaine d'Intérêt Majeur, Réseau Francilien en Sciences Informatiques*).

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

- Master: Bruno Blanchet, Cryptographic protocols: formal and computational proofs, 18h equivalent TD, master M2 MPRI, université Paris VII, France
- Master: Karthikeyan Bhargavan, Cryptographic protocols: formal and computational proofs, 18h equivalent TD, master M2 MPRI, université Paris VII, France

- Master: Karthikeyan Bhargavan, Network Protocol Safety and Security, 18h equivalent TD, ACN master, Telecom ParisTech
- PhD: Formally Secure Compartmentalizing Compilation course at International School on Foundations of Security Analysis and Design (FOSAD), 27-28 August, 2018, Bertinoro, Italy
- PhD: Program Verification with F* course at EPIT 2018 Software Verification Spring School, 7-11 May, 2018, Aussois, France
- PhD: Attacks and Automated Tools, BIU winter school on cryptography, 11-15 February, 2018, Tel Aviv, Israel
- PhD: Crypto standards for the Internet and Web. ECRYPT-CSA School on Societal Aspects of Cryptology and on Business and Innovation in Crypto. 7-9 January. Zurich, Switzerland.
- PhD: Mix networking, ECRYPT Summer School, ECRYPT-NET School on Integrating Advanced Cryptography with Applications, 16-21 September 2018, Kos, Greece.

9.2.2. Supervision

- PhD: Jean Karim Zinzindohoue, Secure, Fast and Verified Cryptographic Applications: A Scalable Approach [13], ENS Paris, defended on July 3, 2018, supervised by Karthikeyan Bhargavan.
- PhD: Nadim Kobeissi, Formal Verification for Real-World Cryptographic Protocols and Implementations [12], ENS Paris, defended on December 10, 2018, supervised by Karthikeyan Bhargavan and Bruno Blanchet.
- PhD in progress: Benjamin Beurdouche, on verified cryptographic protocol implementations, ENS Paris, since October 2016, supervised by Karthikeyan Bhargavan.
- PhD in progress: Marina Polubelova, on verified post-quantum cryptography, PSL Paris, since October 2017, supervised by Karthikeyan Bhargavan.
- PhD in progress: Natalia Kulatova, on verified secure hardware APIs, PSL Paris, since October 2017, supervised by Karthikeyan Bhargavan.
- PhD in progress: Denis Merigoux, on verified RUST applications, PSL Paris, since November 2017, supervised by Karthikeyan Bhargavan.
- PhD in progress: Benjamin Lipp, On Mechanised Cryptographic Proofs of Protocols and their Link with Verified Implementations, ENS Paris, since October 2018, supervised by Bruno Blanchet and Karthikeyan Bhargavan.
- PhD in progress: Kenji Maillard, on Semantic Foundations for F*, started January 2017, supervised by Catalin Hritcu and Karthikeyan Bhargavan
- PhD in progress: Carmine Abate, The Formal Foundations of Secure Compilation, since June 2018, advised by Catalin Hritcu and Bruno Blanchet
- PhD in progress: Jérémy Thibault, Secure Compartmentalizing Compilation to a Tagged Architecture, from August 2018, advised by Catalin Hritcu and Bruno Blanchet
- PhD in progress: Guido Martínez (CIFASIS-CONICET Rosario), Metatheory for Semi-Automatic Verification of Effectful Programs, from April 2017, advised by Mauro Jaskelioff (CIFASIS-CONICET Rosario) and Catalin Hritcu

9.2.3. Juries

- Karthikeyan Bhargavan participated in the PhD jury of Daniel Fett at University of Stuttgart.
- Harry Halpin participated in the PhD jury of Joseph Raad at University Paris-Saclay.

9.3. Popularization

9.3.1. Internal or external Inria responsibilities

- Bruno Blanchet was co-president of the Inria hiring committee for PhD, post-docs, and *délégations* (*Commision des Emplois Scientifiques*, CES).
- Bruno Blanchet was representative of Inria Paris at the DIM RFSI (*Domaine d'Intérêt Majeur, Réseau Francilien en Sciences Informatiques*).

9.3.2. Interventions

- Karthikeyan Bhargavan was a panelist at the Cloudflare Internet Summit in London, June 14, 2018.
- Harry Halpin was a panelist at the World Digital Asset Summit in San Fransisco, USA, December 10, 2018.

10. Bibliography

Major publications by the team in recent years

[1] M. Abadi, B. Blanchet, C. Fournet. *The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication*, in "Journal of the ACM (JACM)", October 2017, vol. 65, n° 1, pp. 1 - 103 [*DOI :* 10.1145/3127586], https://hal.inria.fr/hal-01636616

[2] C. Abate, A. Azevedo de Amorim, R. Blanco, A. N. Evans, G. Fachini, C. Hriţcu, T. Laurent, B. C. Pierce, M. Stronati, A. Tolmach. *When Good Components Go Bad: Formally Secure Compilation Despite Dynamic Compromise*, in "25th ACM Conference on Computer and Communications Security (CCS)", Toronto, Canada, ACM, October 2018, pp. 1351–1368, https://arxiv.org/abs/1802.00588 [*DOI :* 10.1145/3243734.3243745], https://hal.archives-ouvertes.fr/hal-01949202

[3] A. Azevedo de Amorim, M. Dénès, N. Giannarakis, C. Hriţcu, B. C. Pierce, A. Spector-Zabusky, A. Tolmach. *Micro-Policies: Formally Verified, Tag-Based Security Monitors*, in "36th IEEE Symposium on Security and Privacy (Oakland S&P)", IEEE Computer Society, May 2015, pp. 813–830 [*DOI :* 10.1109/SP.2015.55], https://hal.inria.fr/hal-01265666

[4] K. Bhargavan, B. Blanchet, N. Kobeissi. *Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate*, in "38th IEEE Symposium on Security and Privacy", San Jose, United States, May 2017, pp. 483 - 502 [*DOI :* 10.1109/SP.2017.26], https://hal.inria.fr/hal-01575920

[5] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, P.-Y. Strub. *Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS*, in "IEEE Symposium on Security and Privacy (Oakland)", 2014, pp. 98–113, https://hal.inria.fr/hal-01102259

[6] B. Blanchet. *Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif*, in "Foundations and Trends in Privacy and Security", October 2016, vol. 1, n° 1–2, pp. 1–135, https://hal.inria.fr/hal-01423760

[7] M. Isaakidis, H. Halpin, G. Danezis. *UnlimitID: Privacy-Preserving Federated Identity Management Using Algebraic MACs*, in "Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society", New York, NY, USA, WPES '16, ACM, 2016, pp. 139–142 [*DOI :* 10.1145/2994620.2994637], https://hal.inria.fr/hal-01426847

[8] N. Kobeissi, K. Bhargavan, B. Blanchet. *Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach*, in "2nd IEEE European Symposium on Security and Privacy", Paris, France, April 2017, pp. 435 - 450 [*DOI :* 10.1109/EuroSP.2017.38], https://hal.inria.fr/hal-01575923

[9] N. SWAMY, C. HRITCU, C. KELLER, A. RASTOGI, A. DELIGNAT-LAVAUD, S. FOREST, K. BHARGAVAN, C. FOURNET, P.-Y. STRUB, M. KOHLWEISS, J.-K. ZINZINDOHOUÉ, S. ZANELLA-BÉGUELIN. *Dependent Types and Multi-Monadic Effects in F**, in "43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", ACM, January 2016, pp. 256-270, https://hal.inria.fr/hal-01265793

[10] J.-K. ZINZINDOHOUÉ, K. BHARGAVAN, J. PROTZENKO, B. BEURDOUCHE. *HACL*: A Verified Modern Cryptographic Library*, in "Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017", 2017, pp. 1789–1806, https://hal.inria.fr/hal-01588421

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] C. HRIŢCU. *The Quest for Formally Secure Compartmentalizing Compilation*, ENS Paris ; PSL Research University, January 2019, Habilitation à diriger des recherches, https://tel.archives-ouvertes.fr/tel-01995823

[12] N. KOBEISSI. *Formal Verification for Real-World Cryptographic Protocols and Implementations*, Inria Paris ; Ecole Normale Supérieure de Paris - ENS Paris, December 2018, https://hal.inria.fr/tel-01950884

[13] J.-K. ZINZINDOHOUÉ. *Secure, fast and verified cryptographic applications: a scalable approach*, Université de recherche Paris Sciences Lettres – PSL Research University, July 2018, https://hal.inria.fr/tel-01981380

Articles in International Peer-Reviewed Journals

[14] D. ADRIAN, K. BHARGAVAN, Z. DURUMERIC, P. GAUDRY, M. GREEN, J. A. HALDERMAN, N. HENINGER, D. SPRINGALL, E. THOMÉ, L. VALENTA, B. VANDERSLOOT, E. WUSTROW, S. ZANELLA-BÉGUELIN, P. ZIMMERMANN. *Imperfect forward secrecy: How Diffie-Hellman fails in practice*, in "Communications of the ACM", December 2018, vol. 62, n$^{\mathrm{o}}$ 1, pp. 106-114 [*DOI :* 10.1145/3292035], https://hal.inria.fr/hal-01982426

[15] D. AHMAN. *Handling Fibred Algebraic Effects*, in "Proceedings of the ACM on Programming Languages", January 2018, vol. 2, n$^{\mathrm{o}}$ POPL [*DOI :* 10.1145/3158095], https://hal.archives-ouvertes.fr/hal-01672734

[16] D. AHMAN, C. FOURNET, C. HRIŢCU, K. MAILLARD, A. RASTOGI, N. SWAMY. *Recalling a Witness: Foundations and Applications of Monotonic State*, in "Proceedings of the ACM on Programming Languages", January 2018, vol. 2, n$^{\mathrm{o}}$ POPL, https://arxiv.org/abs/1707.02466 [*DOI :* 10.1145/3158153], https://hal.archives-ouvertes.fr/hal-01672733

[17] B. BLANCHET, B. SMYTH. *Automated reasoning for equivalences in the applied pi calculus with barriers*, in "Journal of Computer Security", 2018, vol. 26, n$^{\mathrm{o}}$ 3, pp. 367 - 422 [*DOI :* 10.3233/JCS-171013], https://hal.inria.fr/hal-01947972

[18] W. J. BOWMAN, Y. CONG, N. RIOUX, A. AHMED. *Type-Preserving CPS Translation of Σ and Π Types is Not Not Possible*, in "Proceedings of the ACM on Programming Languages", January 2018, vol. 2, n$^{\mathrm{o}}$ POPL [*DOI :* 10.1145/3158110], https://hal.archives-ouvertes.fr/hal-01672735

[19] O. FLÜCKIGER, G. SCHERER, M.-H. YEE, A. GOEL, A. AHMED, J. VITEK. *Correctness of Speculative Optimizations with Dynamic Deoptimization*, in "Proceedings of the ACM on Programming Languages",

2018, vol. 2, n^O^ POPL, https://arxiv.org/abs/1711.03050 [*DOI :* 10.1145/3158137], https://hal.inria.fr/hal-01646765

[20] M. NEW, A. AHMED. *Graduality from embedding-projection pairs*, in "Proceedings of the ACM on Programming Languages", July 2018, vol. 2, n^O^ ICFP, pp. 1-30, https://arxiv.org/abs/1807.02786 [*DOI :* 10.1145/3236768], https://hal.archives-ouvertes.fr/hal-01949209

[21] N. TABAREAU, É. TANTER, M. SOZEAU. *Equivalences for Free: Univalent Parametricity for Effective Transport*, in "Proceedings of the ACM on Programming Languages", September 2018, pp. 1-29 [*DOI :* 10.1145/3234615], https://hal.inria.fr/hal-01559073

[22] M. TORO, R. GARCIA, É. TANTER. *Type-Driven Gradual Security with References*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", December 2018, vol. 40, n^O^ 4, pp. 1-55 [*DOI :* 10.1145/3229061], https://hal.archives-ouvertes.fr/hal-01957581

[23] M. TORO, E. LABRADA, É. TANTER. *Gradual Parametricity, Revisited*, in "Proceedings of the ACM on Programming Languages", 2018, vol. 3, n^O^ POPL, https://arxiv.org/abs/1807.04596 [*DOI :* 10.1145/3290330], https://hal.archives-ouvertes.fr/hal-01960553

[24] N. VAZOU, É. TANTER, D. VAN HORN. *Gradual liquid type inference*, in "Proceedings of the ACM on Programming Languages", October 2018, vol. 2, n^O^ OOPSLA, pp. 1-25, https://arxiv.org/abs/1807.02132 [*DOI :* 10.1145/3276502], https://hal.archives-ouvertes.fr/hal-01949207

International Conferences with Proceedings

[25] C. ABATE, A. AZEVEDO DE AMORIM, R. BLANCO, A. N. EVANS, G. FACHINI, C. HRIŢCU, T. LAURENT, B. C. PIERCE, M. STRONATI, A. TOLMACH. *When Good Components Go Bad: Formally Secure Compilation Despite Dynamic Compromise*, in "25th ACM Conference on Computer and Communications Security (CCS)", Toronto, Canada, ACM, October 2018, pp. 1351–1368, https://arxiv.org/abs/1802.00588 [*DOI :* 10.1145/3243734.3243745], https://hal.archives-ouvertes.fr/hal-01949202

[26] A. AZEVEDO DE AMORIM, C. HRIŢCU, B. C. PIERCE. *The Meaning of Memory Safety*, in "7th International Conference on Principles of Security and Trust (POST)", Thessaloniki, Greece, April 2018, pp. 79–105, https://arxiv.org/abs/1705.07354 [*DOI :* 10.1007/978-3-319-89722-6_4], https://hal.archives-ouvertes.fr/hal-01949201

[27] D. BAELDE, A. LICK, S. SCHMITZ. *A Hypersequent Calculus with Clusters for Linear Frames*, in "Twefth Conference on Advances in Modal Logic", Bern, Switzerland, G. BEZHANISHVILI, G. D'AGOSTINO, G. METCALFE, T. STUDE (editors), Advances in Modal Logic, College Publications, July 2018, vol. 12, pp. 36–55, https://hal.inria.fr/hal-01756126

[28] D. BAELDE, A. LICK, S. SCHMITZ. *A Hypersequent Calculus with Clusters for Tense Logic over Ordinals*, in "38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science", Ahmedabad, India, S. GANGULY, P. PANDYA (editors), Leibniz International Proceedings in Informatics, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, vol. 122, pp. 15:1–15:19 [*DOI :* 10.4230/LIPIcs.FSTTCS.2018.15], https://hal.inria.fr/hal-01852077

[29] K. BHARGAVAN, I. BOUREANU, A. DELIGNAT-LAVAUD, P.-A. FOUQUE, C. ONETE. *A Formal Treatment of Accountable Proxying over TLS*, in "SP 2018 - IEEE Symposium on Security and Privacy", San Francisco, United States, May 2018, https://hal.inria.fr/hal-01948722

[30] K. BHARGAVAN, F. KIEFER, P.-Y. STRUB. *hacspec: Towards Verifiable Crypto Standards*, in "Security Standardisation Research. SSR 2018", Darmstadt, Germany, November 2018, pp. 1-20 [*DOI :* 10.1007/978-3-030-04762-7_1], https://hal.inria.fr/hal-01967342

[31] B. BLANCHET. *Composition Theorems for CryptoVerif and Application to TLS 1.3*, in "31st IEEE Computer Security Foundations Symposium (CSF'18)", Oxford, United Kingdom, July 2018 [*DOI :* 10.1109/CSF.2018.00009], https://hal.inria.fr/hal-01947959

[32] W. BOWMAN, A. AHMED. *Typed closure conversion for the calculus of constructions*, in "PLDI'18 - 39th ACM SIGPLAN Conference on Programming Language Design and Implementation", Philadelphia, PA, United States, June 2018, https://arxiv.org/abs/1808.04006 [*DOI :* 10.1145/3296979.3192372], https://hal.archives-ouvertes.fr/hal-01949211

[33] N. GRIMM, K. MAILLARD, C. FOURNET, C. HRIŢCU, M. MAFFEI, J. PROTZENKO, T. RAMANANANDRO, A. RASTOGI, N. SWAMY, S. ZANELLA-BÉGUELIN. *A Monadic Framework for Relational Verification: Applied to Information Security, Program Equivalence, and Optimizations*, in "7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)", Los Angeles, United States, ACM, January 2018, pp. 130–145, https://arxiv.org/abs/1703.00055 [*DOI :* 10.1145/3167090], https://hal.archives-ouvertes.fr/hal-01672703

[34] N. KOBEISSI, N. KULATOVA. *Ledger Design Language: Designing and Deploying Formally Verified Public Ledgers*, in "Workshop on Security Protocol Implementations: Development and Analysis", London, United Kingdom, April 2018, https://hal.inria.fr/hal-01948971

[35] G. SCHERER, M. NEW, N. RIOUX, A. AHMED. *FabULous Interoperability for ML and a Linear Language*, in "International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)", Thessaloniki, Greece, C. BAIE, U. D. LAGO (editors), FabOpen image in new windowous Interoperability for ML and a Linear Language, Springer, April 2018, vol. LNCS - Lecture Notes in Computer Science, n^{o} 10803, https://arxiv.org/abs/1707.04984 [*DOI :* 10.1007/978-3-319-89366-2_8], https://hal.inria.fr/hal-01929158

Conferences without Proceedings

[36] N. KOBEISSI, K. BHARGAVAN. *Noise Explorer: Fully Automated Modeling and Verification for Arbitrary Noise Protocols*, in "RWC 2019 - Real World Cryptography Symposium", San Jose, United States, January 2019, https://hal.inria.fr/hal-01948964

[37] K. MAILLARD, É. MIQUEY, X. MONTILLET, G. MUNCH-MACCAGNONI, G. SCHERER. *A preview of a tutorial on L (polarized $\mu\mu$-tilde)*, in "HOPE 2018 - 7th ACM SIGPLAN Workshop on Higher-Order Programming with Effects", St. Louis, United States, September 2018, https://hal.inria.fr/hal-01992294

[38] G. MARTÍNEZ, D. AHMAN, V. DUMITRESCU, N. GIANNARAKIS, C. HAWBLITZEL, C. HRIŢCU, M. NARASIMHAMURTHY, Z. PARASKEVOPOULOU, C. PIT-CLAUDEL, J. PROTZENKO, T. RAMANANANDRO, A. RASTOGI, N. SWAMY. *Meta-F*: Proof automation with SMT, Tactics, and Metaprograms*, in "ESOP'19 - European Symposium on Programming", Prague, Czech Republic, April 2019, https://arxiv.org/abs/1803.06547 , https://hal.archives-ouvertes.fr/hal-01995376

Research Reports

[39] B. BLANCHET. *Composition Theorems for CryptoVerif and Application to TLS 1.3*, Inria Paris, April 2018, n^{o} RR-9171, 67 p. , https://hal.inria.fr/hal-01764527

Other Publications

[40] D. BAELDE, A. LICK, S. SCHMITZ. *Decidable XPath Fragments in the Real World*, August 2018, working paper or preprint, https://hal.inria.fr/hal-01852475

[41] H. HALPIN, K. ERMOSHINA, F. MUSIANI. *Co-ordinating Developers and High-Risk Users of Privacy-Enhanced Secure Messaging Protocols*, November 2018, SSR 2018 - Security Standardisation Research Conference, https://hal.inria.fr/hal-01966560

[42] H. HALPIN. *Decentralizing the Social Web*, October 2018, INSCI'2018- 5th International conference 'Internet Science', https://hal.inria.fr/hal-01966561

[43] N. KOBEISSI. *Capsule: A Protocol for Secure Collaborative Document Editing*, December 2018, working paper or preprint, https://hal.inria.fr/hal-01948967

References in notes

[44] M. ABADI, B. BLANCHET. *Analyzing Security Protocols with Secrecy Types and Logic Programs*, in "Journal of the ACM", January 2005, vol. 52, n^{o} 1, pp. 102–146, http://prosecco.gforge.inria.fr/personal/bblanche/publications/AbadiBlanchetJACM7037.pdf

[45] M. ABADI, B. BLANCHET, C. FOURNET. *Just Fast Keying in the Pi Calculus*, in "ACM Transactions on Information and System Security (TISSEC)", July 2007, vol. 10, n^{o} 3, pp. 1–59, http://prosecco.gforge.inria.fr/personal/bblanche/publications/AbadiBlanchetFournetTISSEC07.pdf

[46] C. ABATE, R. BLANCO, D. GARG, C. HRITCU, M. PATRIGNANI, J. THIBAULT. *Journey Beyond Full Abstraction: Exploring Robust Property Preservation for Secure Compilation*, July 2018, arXiv:1807.04603, https://arxiv.org/abs/1807.04603

[47] C. ABATE, A. AZEVEDO DE AMORIM, R. BLANCO, A. N. EVANS, G. FACHINI, C. HRITCU, T. LAURENT, B. C. PIERCE, M. STRONATI, A. TOLMACH. *When Good Components Go Bad: Formally Secure Compilation Despite Dynamic Compromise*, in "25th ACM Conference on Computer and Communications Security (CCS)", ACM, October 2018, pp. 1351–1368, https://arxiv.org/abs/1802.00588

[48] D. AHMAN, C. FOURNET, C. HRITCU, K. MAILLARD, A. RASTOGI, N. SWAMY. *Recalling a Witness: Foundations and Applications of Monotonic State*, in "PACMPL", January 2018, vol. 2, n^{o} POPL, pp. 65:1–65:30, https://arxiv.org/abs/1707.02466

[49] D. AHMAN, C. HRIŢCU, K. MAILLARD, G. MARTÍNEZ, G. PLOTKIN, J. PROTZENKO, A. RASTOGI, N. SWAMY. *Dijkstra Monads for Free*, in "44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)", ACM, January 2017, pp. 515-529 [*DOI :* 10.1145/3009837.3009878], https://www.fstar-lang.org/papers/dm4free/

[50] A. AZEVEDO DE AMORIM, M. DÉNÈS, N. GIANNARAKIS, C. HRIŢCU, B. C. PIERCE, A. SPECTOR-ZABUSKY, A. TOLMACH. *Micro-Policies: Formally Verified, Tag-Based Security Monitors*, in "36th IEEE Symposium on Security and Privacy (Oakland S&P)", IEEE Computer Society, May 2015, pp. 813–830 [*DOI :* 10.1109/SP.2015.55], http://prosecco.gforge.inria.fr/personal/hritcu/publications/micro-policies.pdf

[51] A. AZEVEDO DE AMORIM, C. HRITCU, B. C. PIERCE. *The Meaning of Memory Safety*, in "7th International Conference on Principles of Security and Trust (POST)", April 2018, pp. 79–105 [*DOI :* 10.1007/978-3-319-89722-6_4], https://arxiv.org/abs/1705.07354

[52] K. BHARGAVAN, B. BLANCHET, N. KOBEISSI. *Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate*, in "38th IEEE Symposium on Security and Privacy", San Jose, United States, May 2017, pp. 483 - 502 [*DOI :* 10.1109/SP.2017.26], https://hal.inria.fr/hal-01575920

[53] K. BHARGAVAN, B. BOND, A. DELIGNAT-LAVAUD, C. FOURNET, C. HAWBLITZEL, C. HRITCU, S. ISHTIAQ, M. KOHLWEISS, R. LEINO, J. LORCH, K. MAILLARD, J. PANG, B. PARNO, J. PROTZENKO, T. RAMANANANDRO, A. RANE, A. RASTOGI, N. SWAMY, L. THOMPSON, P. WANG, S. ZANELLA-BÉGUELIN, J.-K. ZINZINDOHOUÉ. *Everest: Towards a Verified, Drop-in Replacement of HTTPS*, in "2nd Summit on Advances in Programming Languages (SNAPL)", May 2017, http://drops.dagstuhl.de/opus/volltexte/2017/7119/pdf/LIPIcs-SNAPL-2017-1.pdf

[54] K. BHARGAVAN, A. DELIGNAT-LAVAUD, C. FOURNET, M. KOHLWEISS, J. PAN, J. PROTZENKO, A. RASTOGI, N. SWAMY, S. ZANELLA-BÉGUELIN, J.-K. ZINZINDOHOUÉ. *Implementing and Proving the TLS 1.3 Record Layer*, in "IEEE Symposium on Security and Privacy (Oakland)", 2017

[55] K. BHARGAVAN, C. FOURNET, R. CORIN, E. ZALINESCU. *Verified Cryptographic Implementations for TLS*, in "ACM Transactions Inf. Syst. Secur.", March 2012, vol. 15, n^{o} 1, pp. 3:1–3:32, http://doi.acm.org/10.1145/2133375.2133378

[56] K. BHARGAVAN, C. FOURNET, A. D. GORDON, N. SWAMY. *Verified implementations of the information card federated identity-management protocol*, in "ACM Symposium on Information, Computer and Communications Security (ASIACCS)", 2008, pp. 123-135

[57] B. BLANCHET, M. ABADI, C. FOURNET. *Automated Verification of Selected Equivalences for Security Protocols*, in "Journal of Logic and Algebraic Programming", February–March 2008, vol. 75, n^{o} 1, pp. 3–51, http://prosecco.gforge.inria.fr/personal/bblanche/publications/BlanchetAbadiFournetJLAP07.pdf

[58] B. BLANCHET. *An Efficient Cryptographic Protocol Verifier Based on Prolog Rules*, in "14th IEEE Computer Security Foundations Workshop (CSFW'01)", 2001, pp. 82–96

[59] B. BLANCHET. *Automatic Verification of Correspondences for Security Protocols*, in "Journal of Computer Security", July 2009, vol. 17, n^{o} 4, pp. 363–434, http://prosecco.gforge.inria.fr/personal/bblanche/publications/BlanchetJCS08.pdf

[60] B. BLANCHET, A. PODELSKI. *Verification of Cryptographic Protocols: Tagging Enforces Termination*, in "Theoretical Computer Science", March 2005, vol. 333, n^{o} 1-2, pp. 67–90, Special issue FoSSaCS'03, http://prosecco.gforge.inria.fr/personal/bblanche/publications/BlanchetPodelskiTCS04.html

[61] D. CADÉ, B. BLANCHET. *Proved Generation of Implementations from Computationally Secure Protocol Specifications*, in "Journal of Computer Security", 2015, vol. 23, n^{o} 3, pp. 331–402

[62] J. CLULOW. *On the Security of PKCS#11*, in "CHES", 2003, pp. 411-425

[63] S. DELAUNE, S. KREMER, G. STEEL. *Formal Analysis of PKCS#11 and Proprietary Extensions*, in "Journal of Computer Security", November 2010, vol. 18, n^{o} 6, pp. 1211-1245 [*DOI :* 10.3233/JCS-2009-0394], http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKS-jcs09.pdf

[64] A. DELIGNAT-LAVAUD, K. BHARGAVAN, S. MAFFEIS. *Language-Based Defenses Against Untrusted Browser Origins*, in "Proceedings of the 22th USENIX Security Symposium", 2013, http://prosecco.inria.fr/personal/karthik/pubs/language-based-defenses-against-untrusted-origins-sec13.pdf

[65] D. DOLEV, A. YAO. *On the security of public key protocols*, in "IEEE Transactions on Information Theory", 1983, vol. IT–29, n^{o} 2, pp. 198–208

[66] C. FOURNET, M. KOHLWEISS, P.-Y. STRUB. *Modular Code-Based Cryptographic Verification*, in "ACM Conference on Computer and Communications Security", 2011

[67] N. GRIMM, K. MAILLARD, C. FOURNET, C. HRITCU, M. MAFFEI, J. PROTZENKO, T. RAMANANANDRO, A. RASTOGI, N. SWAMY, S. ZANELLA-BÉGUELIN. *A Monadic Framework for Relational Verification: Applied to Information Security, Program Equivalence, and Optimizations*, in "7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)", ACM, January 2018, pp. 130–145 [*DOI :* 10.1145/3167090], https://arxiv.org/abs/1703.00055

[68] N. KOBEISSI, K. BHARGAVAN, B. BLANCHET. *Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach*, in "2nd IEEE European Symposium on Security and Privacy", Paris, France, April 2017, pp. 435 - 450 [*DOI :* 10.1109/EUROSP.2017.38], https://hal.inria.fr/hal-01575923

[69] G. MARTÍNEZ, D. AHMAN, V. DUMITRESCU, N. GIANNARAKIS, C. HAWBLITZEL, C. HRITCU, M. NARASIMHAMURTHY, Z. PARASKEVOPOULOU, C. PIT-CLAUDEL, J. PROTZENKO, T. RAMANANANDRO, A. RASTOGI, N. SWAMY. *Meta-F*: Proof Automation with SMT, Tactics, and Metaprograms*, March 2018, arXiv:1803.06547, https://arxiv.org/abs/1803.06547

[70] R. NEEDHAM, M. SCHROEDER. *Using encryption for authentication in large networks of computers*, in "Communications of the ACM", 1978, vol. 21, n^{o} 12, pp. 993–999

[71] J. PROTZENKO, J.-K. ZINZINDOHOUÉ, A. RASTOGI, T. RAMANANANDRO, P. WANG, S. ZANELLA-BÉGUELIN, A. DELIGNAT-LAVAUD, C. HRITCU, K. BHARGAVAN, C. FOURNET, N. SWAMY. *Verified Low-Level Programming Embedded in F**, in "PACMPL", September 2017, vol. 1, n^{o} ICFP, pp. 17:1–17:29 [*DOI :* 10.1145/3110261], http://arxiv.org/abs/1703.00053

[72] N. SWAMY, C. FOURNET, A. RASTOGI, K. BHARGAVAN, J. CHEN, P.-Y. STRUB, G. M. BIERMAN. *Gradual typing embedded securely in JavaScript*, in "41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", 2014, pp. 425-438, http://prosecco.inria.fr/personal/karthik/pubs/tsstar-popl14.pdf

[73] N. SWAMY, C. HRIŢCU, C. KELLER, A. RASTOGI, A. DELIGNAT-LAVAUD, S. FOREST, K. BHARGAVAN, C. FOURNET, P.-Y. STRUB, M. KOHLWEISS, J.-K. ZINZINDOHOUÉ, S. ZANELLA-BÉGUELIN. *Dependent Types and Multi-Monadic Effects in F**, in "43rd ACM SIGPLAN-SIGACT Symposium on Principles of

Programming Languages (POPL)", ACM, January 2016, pp. 256-270, https://www.fstar-lang.org/papers/mumon/

[74] J.-K. ZINZINDOHOUÉ, K. BHARGAVAN, J. PROTZENKO, B. BEURDOUCHE. *HACL*: A Verified Modern Cryptographic Library*, in "Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017", 2017, pp. 1789–1806, http://doi.acm.org/10.1145/3133956.3134043